[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/847,642	05/01/2001	Luciano Lavagno	261/246	6620

23639 7590 05/16/2005

BINGHAM, MCCUTCHEN LLP
THREE EMBARCADERO CENTER
18 FLOOR
SAN FRANCISCO, CA 94111-4067

EXAMINER	
GUILL, RUSSELL L	
ART UNIT	PAPER NUMBER
2123	

DATE MAILED: 05/16/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	**Application No.** 09/847,642	**Applicant(s)** LAVAGNO ET AL.	
	Examiner Russell L. Guill	**Art Unit** 2123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 May 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☐ Claim(s) _____ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1 - 60* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *07/27/2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date ~~[illegible]~~ 2 pages

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

4

DETAILED ACTION

1. Claims 1 – 60 have been examined. Claims 1 – 60 have been rejected.

Claim Objections

2. Claim 29 objected to because of the following informalities: The word "assembler" is spelled "assember". Appropriate correction is required.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 1 – 13, 16 – 19, 21 – 22, 33 – 45, 47 - 51, 53 are rejected under 35 U.S.C. 102(b) as being anticipated by Lajolo (Lajolo, Marcello; Lazarescu, Mihai; Sangiovanni-Vincentelli, Alberto; "A Compilation-based Software Estimation Scheme for Hardware/Software Co-Simulation", 7th International Workshop on Hardware/Software Co-Design, May 3 – 5, 1999).

4.1. Regarding claims 1 and 33, Lajolo appears to teach a method of performing software performance analysis for a target machine (***page 85, Abstract***) comprising:

4.1.1. Describing a system design as a network of logical entities (***page 86 section 3 The integration of the GCC suite in the POLIS framework, paragraph 2; and page 87, figure 1, labeled The Simulation Flow***);

4.1.2. Selecting at least one of the logical entities for a software implementation (***page 86 section 3 The integration of the GCC suite in the POLIS framework, paragraph 4; and page 87, figure 1, labeled The Simulation Flow***);

4.1.3. Synthesizing a software program from the logical entities selected for the software implementation (***page 87, figure 1 labeled The simulation flow, element SW synthesis***);

4.1.4. Compiling the software program to generate an optimized assembler code representation of the software program (***page 87, figure 1 labeled The simulation flow, element Compilation and element Optimization options***);

4.1.5. Performing a performance analysis using the assembler code (***page 87, figure 1 labeled The simulation flow, element Instruction level timing analysis***);

4.1.6. Generating a software simulation model using the assembler code (***page 87, figure 1 labeled The simulation flow, element C simulation model***);

4.1.7. Generating a hardware/software co-simulation model using the simulation model (***page 87, figure 1 labeled The simulation flow, element HW/SW co-simulation***).

4.2. Regarding claims 2 and 34, Lajolo appears to teach that the compiling step further comprises incorporating a description of the target machine (***page 87, figure 1 labeled The simulation flow, element ,md description; and page 88, section 3.1, first paragraph***).

4.3. Regarding claims 3 and 35, Lajolo appears to teach that the software simulation model is an assembler-level C code simulation model (***page 87, figure 2, bottom code list***).

4.4. Regarding claims 4 and 36, Lajolo appears to teach selecting at least one of the logical entities for a hardware implementation, and synthesizing a software model of the hardware implementation from the selected logical entities, wherein the hardware/software co-simulation model is generated using the software model of the hardware implementation (***page 87, figure 1 labeled The simulation flow, elements HW/SW partitioning, and HW C model, and HW/SW co-simulation***).

4.5. Regarding claims 5 and 37, Lajolo appears to teach that the performance analysis measures an execution time of an element of the assembler code (***page 87, figure 1 labeled The simulation flow, element Instruction level timing analysis***).

4.6. Regarding claims 6 and 38, Lajolo appears to teach that the software program is compiled using the same compiler used to compile a production executable (***page 88, section 3.3, paragraph 1, item 1 that starts with "1. if GCC is a suitable . . . "***).

4.7. Regarding claims 7 and 39, Lajolo appears to teach that that performing the performance analysis comprises annotating the assembler code with performance information (***page 87, figure 2; and page 87, right-side column, the paragraphs below figure 2***).

4.8. Regarding claims 8 and 40, Lajolo appears to teach that that the performance information is timing information (***page 87, figure 2; and page 87, right-side column, the paragraphs below figure 2***).

4.9. Regarding claims 9 and 41, Lajolo appears to teach preparing software for a performance estimation comprising:

4.9.1. Providing a software assembly code model (***page 87, figure 2, the middle partition is assembly code; and page 87, right-side column, paragraphs below figure 2***);

4.9.2. Translating the assembly code module into a simulation model (***page 87, figure 2, the bottom partition is a simulation model; and page 87, right-side column, paragraphs below figure 2***);

4.9.3. Annotating the simulation model with performance information (***page 87, figure 2, the bottom partition is a simulation model; and page 87, right-side column, paragraphs below figure 2***);

4.10. Regarding claims 10 and 42, Lajolo appears to teach that providing a software assembly code module comprises compiling software source code to assembly (***page 87, figure 2, the top partition is software source code and the middle partition is an assembly code module page 87, right-side column, paragraphs below figure 2***);

4.11. Regarding claims 11 and 43, Lajolo appears to teach that the software assembly code module is compiled using a compiler adapted to create code that will execute on a first machine architecture (***page 87, figure 2, middle and bottom partitions; and right-side column, paragraphs below figure 2; please note that the compile produces C code from the assembly code***).

4.12. Regarding claims 12 and 44, Lajolo appears to teach that the performance information is associated with the first machine architecture (***page 87, figure 1, element instruction level timing analysis***).

4.13. Regarding claims 13 and 45, Lajolo appears to teach that the simulation model is compiled to execute on a second machine architecture, the second machine architecture being different from the first machine architecture (**page 88, left-side column, first paragraph**).

4.14. Regarding claims 15 and 47, Lajolo appears to teach that the simulation model is an assembler-level representation of the software, expressed in a high-level language (***page 87, figure 2, bottom partition***).

4.15. Regarding claims 16 and 48, Lajolo appears to teach that the translation step gathers information from another software module (***page 87, figure 1, element .md description; and page 88, figure 3***).

4.16. Regarding claims 17 and 49, Lajolo appears to teach that information gathered comprises high-level hints about software assembly code module (***page 88, left-side column, first paragraph (delays in the .md file)***).

4.17. Regarding claims 18 and 50, Lajolo appears to teach that the performance information comprises estimated performance information (***page 87, left-side column, paragraph 3, re: timing information***).

4.18. Regarding claims 19 and 51, Lajolo appears to teach that the performance information is statically estimated (***page 87, left-side column, paragraph 3, re: timing information***).

4.19. Regarding claims 21 and 53, Lajolo appears to teach that compiling the simulation model to a simulator host program (***page 88, left-side column, first paragraph***); and executing the simulator host program on a simulator to allow performance measurements to be taken (***page 89, table 2 and table 3, and section 4 Performance Simulation and Results***).

4.20. Regarding claims 22 and 54, Lajolo appears to teach linking an already annotated module with the simulation model (***page 87, figure 1, element HW C model; and page 87, left-side column, last paragraph continued at the top of the right-side column; and page 87, figure 2, bottom partition of annotated C code***).

4.21. Regarding claims 33 and 41, the performance results were executed on a SUN SPARC-20 workstation which would typically have had a computer program product that includes a medium useable by a processor containing a sequence of instructions for the processor to execute (***page 89, left-side column, paragraph 4***).

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 23 and 55 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lajolo (Lajolo, Marcello; Lazarescu, Mihai; Sangiovanni-Vincentelli, Alberto; "A Compilation-based Software Estimation Scheme for Hardware/Software Co-Simulation", 7th International Workshop on Hardware/Software Co-Design, May 3 – 5, 1999) in view of Mak (Mak,Ronald; "Writing Compilers and Interpreters: an applied approach", 1991, John Wiley & Sons).

6.1. The art of Lajolo is directed toward compilation-based software estimation scheme for hardware/software co-simulation (Title).

6.2. The art of Mak is directed to compilation and interpreting software programs (Title).

6.3. Regarding claims 23 and 55, Lajolo appears to teach a method:

6.3.1. Receiving the assembly language software module (***page 87, figure 1, element Instruction level timing analysis, and page 87, figure 2, middle partition***);

6.3.2. Processing a data structure to refine the accuracy of the simulation model (***page 87, figure 1, element Instruction level timing analysis)***;

6.3.2.1. Regarding (***page 87, figure 1, element Instruction level timing analysis)***; it would have been obvious that the instructions are in a data structure, and that they are processed to add timing information for accuracy of simulation.

6.3.3. Associating performance information with an element of the assembly language software module (***page 87, figure 1, element Instruction level timing analysis; and page 87, figure 2, middle and bottom partition)***;

6.3.4. Outputting the simulation model (***page 87, figure 1, element C simulation model; and page 87, figure 2, bottom partition***).

6.4. Regarding claims 23 and 55, Lajolo appears to teach using a mapping definition to map each assembler instruction to a period of time (***page 87, figure 1, elements .md description and Instruction level timing analysis; and page 88, left-side column, first paragraph, regarding .md file***).

6.5. Regarding claims 23 and 55, Lajolo does not specifically teach ***parsing the assembly language software module into a data structure, the data structure comprising one or more nodes*** being mapped to a period of time using a mapping definition, ***each of the one or more nodes containing an element of the assembly language software module***.

6.6. Regarding claims 23 and 55, Mak appears to teach parsing the assembly language software module into a data structure comprising one or more nodes being mapped to a period of time, each of the one or more nodes containing an element of the assembly language software module (***page 174, paragraph 3 that starts with "The prime motivation . . "; and page 382, section 15.4, esp. "The intermediate form had a simple linear structure that was interspersed with address markers"; and page 283, chapter 12, "Emitting 8086 Assembly Language Code***).

6.6.1. Regarding (***page 174, paragraph 3 that starts with "The prime motivation . . "; and page 382, section 15.4, esp. "The intermediate form had a simple linear structure that was interspersed with address markers"; and page 283, chapter 12, "Emitting 8086 Assembly Language Code***); it would have been obvious that a linear structure is comprised of nodes, and that each node is interpreted which consumes a period of time. It would have been obvious to build the structure with assembly code.

6.7. The art of Mak and the art of Lajolo are analogous art because they both contain the problem of compiling software modules (***Mak, Title of book; Lajolo, page 87, figure 1***).

6.8. The motivation to use the art of Mak with the art of Lajolo would have been obvious given the benefits recited in Mak that an intermediate form structure gives the compiler an opportunity to analyze the program in order to generate better code (***page 382, section 15.4, second paragraph***), and the compiler in Lajolo performs optimization also (***page 87, figure 1***).

6.9. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Mak and Lajolo to produce the claimed invention.

7. Claims 24 – 26, 28, 56 – 58, and 60 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lajolo and Mak as addressed in claims 23 and 55 above.

7.1. Regarding claims 25 and 57, Lajolo appears to teach that performance information comprises an execution delay value for the element of the assembly language software module (***page 87, figure 2; and page 87, paragraphs directly beneath figure 2***).

7.2. Regarding claims 26 and 58, Lajolo appears to teach that performance information is a statically computed value (***page 88, left-side column, first paragraph***).

7.3. Regarding claims 24 and 56, Lajolo does not specifically teach a data structure wherein the one or more nodes comprises a first node and a second node, the first node being mapped to a first period of time, the second node being mapped to a second period of time, the first period of time being different from the second period of time.

7.4. Regarding claims 28 and 60, Lajolo does not specifically teach that processing the data structure comprises replicating the behavior of the assembly language software model in the simulation model.

7.5. Regarding claims 24 and 56, Mak appears to teach a linear structure that obviously is a series of nodes where each node is an operation that consumes a period of time (***page 382, section 15.4***).

7.6. Regarding claims 28 and 60, Mak appears to teach that processing the data structure comprises replicating the behavior of the assembly language software model in the simulation model (***page 382, section 15.4, first two sentences***).

7.6.1. Regarding (***page 382, section 15.4, first two sentences***); since the interpreter processes the data structure of the linear structure by executing the actions which replicate the behavior of the source software, it would have been obvious to process the data structure to replicate the behavior of the assembly language software model in the simulation model.

8. Claims 29 - 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lajolo (Lajolo, Marcello; Lazarescu, Mihai; Sangiovanni-Vincentelli, Alberto; "A Compilation-based Software Estimation Scheme for Hardware/Software Co-Simulation", 7th International Workshop on Hardware/Software Co-Design, May 3 – 5, 1999) in view of Mak (Mak,Ronald; "Writing Compilers and Interpreters: an applied approach", 1991, John Wiley & Sons).

8.1. The art of Lajolo is directed toward compilation-based software estimation scheme for hardware/software co-simulation (Title).

8.2. The art of Mak is directed to compilation and interpreting software programs (Title).

8.3. The art of Mak and the art of Lajolo are analogous art because they both contain the problem of compiling software modules (***Mak, Title of book; Lajolo, page 87, figure 1***).

8.4. Regarding claim 29:

8.4.1. Lajolo appears to teach:

8.4.1.1. a system for translating assembler code into a simulation model (***page 87, figure 1 and figure 2***).

8.4.1.2. a machine description for providing information about a target machine architecture (***page 88, left-side column, first paragraph; and page 88, figure 3 (.md file); and page 88, section 3.1***).

8.4.1.3. assembler code (***page 87, figure 2***).

8.4.2. Lajolo does not specifically teach:

8.4.2.1. ***A data structure for storing the*** assembler code, ***the data structure comprising one or more nodes, each node representing a unit of time, each of the one or more nodes for storing one or more elements of the*** assembler code;

8.4.2.2. ***One or more operation modules for performing one or more operations on the assembler code stored in the data structure***;

8.4.2.3. ***A storage module for storing information related to the assembler code, the information being used by the one or more operations modules to perform one or more operations on the*** assembler code.

8.4.3. Mak appears to teach:

8.4.3.1. A data structure for storing the program code, the data structure comprising one or more nodes, each node representing a unit of time, each of the one or more nodes for storing one or more elements of the program code (***page 382, section 15.4***);

8.4.3.1.1. Regarding (***page 382, section 15.4***); the linear structure of the intermediate form is obviously a data structure for storing the program code, the data structure comprising one or more nodes, each node representing a unit of time, each of the one or more nodes for storing one or more elements of the program code.

8.4.3.2. One or more operation modules for performing one or more operations on the program code stored in the data structure (***page 199, first paragraph; and page 382, section 15.4, first paragraph***);

8.4.3.2.1. Regarding (***page 199, first paragraph; and page 382, section 15.4, first paragraph***); it would have been obvious that the executor module performs operations on the intermediate form in the linear structure.

8.4.3.3. A storage module for storing information related to the program code, the information being used by the one or more operations modules to perform one or more operations on the program code (***page 46, first paragraph, regarding the symbol table***).

8.4.3.3.1. Regarding (***page 46, first paragraph, regarding the symbol table***); it would have been obvious that the symbol table is a storage module used by the one or more operations modules to perform one or more operations on the program code.

8.5. Regarding claim 30:

8.5.1. Lajolo does not specifically teach that ***the storage module is a symbol table***.

8.5.2. Mak appears to teach that the storage module is a symbol table (***page 46, first paragraph, regarding the symbol table***).

8.6. Regarding claim 31:

8.6.1. Lajolo appears to teach a macro expansion module (***page 87, figure 2, top partition of C code, and middle partition of assembly code***), a simulation model generation module, a simulation model assembly module, and a simulation model export module (***page 87, figure 1, elements Instruction level timing analysis, and C simulation model, and arrow going to HW/SW co-simulation; and page 87, figure 2***).

8.6.1.1. Regarding (***page 87, figure 2, top partition of C code, and middle partition of assembly code***); it would have been obvious to use a macro expansion module to expand the code in the #define statement.

8.6.1.2. Regarding (***page 87, figure 1, elements Instruction level timing analysis, and C simulation model; and page 87, figure 2***); it would have been obvious to have a simulation model generation module, a simulation model assembly module, and a simulation model export module.

8.6.2. Lajolo does not specifically teach that the one or more operations modules comprise a macro expansion module, ***a dependency resolution module, a symbol extraction module, a label identification module***, a simulation model generator module, a simulation model assembly module, and a simulation model export module.

8.6.3. Mak appears to teach that the one or more operations modules comprise a dependency resolution module (***page 383, figure 15-2, tree structure***

demonstrating the dependence of the variable x on the variables a and b), a symbol extraction module (***page 47, middle of the page, Symbol table module, symtab.c, symbol table routines***), and a label identification module (***page 252, section 10.2.5, second paragraph identifies a label element***).

8.6.3.1. Regarding (***page 383, figure 15-2, tree structure demonstrating the dependence of the variable x on the variables a and b***); it would have been obvious to have a module to resolve the dependency of the variable x on variables a and b.

8.6.3.2. Regarding (***page 252, section 10.2.5, second paragraph identifies a label element***); it would have been obvious to have a module to identify the labels and add the label in a symbol table.

8.7. Regarding claim 32:

8.7.1. Lajolo appears to teach that the machine description is provided to an operation module (***page 87, figure 1, elements .md description and Instruction level timing analysis; and page 88, left-side column, first paragraph, "we extracted delays associated in the .md file to each instruction"***).

8.7.1.1. Regarding (***page 87, figure 1, elements .md description and Instruction level timing analysis; and page 88, left-side column, first paragraph, "we extracted delays associated in the .md file to each instruction"***); it would have been obvious that the machine description is provided to an operation module.

8.8. The motivation to use the art of Mak with the art of Lajolo would have been obvious given the benefit of using already designed software architecture in Mak.

8.9. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Mak with the art of Lajolo to produce the claimed inventions.

9. Claims 14 and 46 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lajolo (Lajolo, Marcello; Lazarescu, Mihai; Sangiovanni-Vincentelli, Alberto; "A Compilation-based Software Estimation Scheme for Hardware/Software Co-Simulation", 7th International Workshop on Hardware/Software Co-Design, May 3 – 5, 1999) in view of Hartoog (Hartoog, Mark R.; Rowson, James A.; Reddy, Prakash D.; Desai, Soumya; Dunlop, Douglas D.; Harcourt, Edwin A.; Khullar, Neeti; "Generation of Software Tools from Processor Descriptions for Hardware/Software Codesign", Proceedings of the 34th Design Automation Conference, June 9 – 13 1997).

9.1. Claim 14 is a dependent claim of claim 9, and thereby inherits al of the rejected limitations of claim 9.

9.2. Claim 46 is a dependent claim of claim 41, and thereby inherits al of the rejected limitations of claim 41.

9.3. The art of Lajolo is directed toward compilation-based software estimation scheme for hardware/software co-simulation (***Title***).

9.4. The art of Hartoog is directed toward generation of software tools from processor descriptions for hardware/software codesign (***page 303, Title***).

9.5. Lajolo appears to teach disassembling software binary code (***page 86, left-side column, last paragraph, "generating a C program from the target binary code"***).

9.6. Lajolo does not specifically teach that providing a software assembly code module comprises disassembling software binary code ***to assembly code***.

9.7. Hartoog appears to teach disassembling software binary code to assembly code (***page 305, section 5***).

9.8. The art of Hartoog and the art of Lajolo are analogous art because they both contain the problem of hardware/software codesign (***Lajolo, page 87, figure 1***) and Hartoog (***page 303, title***).

9.9. The motivation to use the art of Hartoog with the art of Lajolo would have been obvious given that in Hartoog, a program's source code may be in C++ (page 304, section 3), and in Lajolo only a C compiler is provided (page 88, section 3.3), so a disassembler would allow the performance estimation to be performed with C++ source code.

10. Claims 20 and 52 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lajolo (Lajolo, Marcello; Lazarescu, Mihai; Sangiovanni-Vincentelli, Alberto; "A Compilation-based Software Estimation Scheme for Hardware/Software Co-Simulation", 7th International Workshop on Hardware/Software Co-Design, May 3 – 5, 1999) in view of Suzuki (Suzuki, Kei; Sangiovanni-Vincentelli, Alberto; "Efficient Software Performance Estimation Methods for Hardware/Software Codesign", 1996, Proceedings of the 33rd annual conference on Design Automation).

10.1. Claim 20 is a dependent claim of claim 9, and thereby inherits all of the rejected limitations of claim 9.

10.2. Claim 52 is a dependent claim of claim 41, and thereby inherits all of the rejected limitations of claim 41.

10.3. The art of Lajolo is directed toward compilation-based software estimation scheme for hardware/software co-simulation (***Title***).

10.4. The art of Suzuki is directed to performance estimation methods for hardware/software codesign (***Title***).

10.5. Regarding claims 20 and 52, Lajolo does not specifically teach that performance information is computed dynamically at run-time, using a formula provided during the annotating step.

10.6. Regarding claims 20 and 52, Suzuki appears to teach that performance information is computed dynamically at run-time, using a formula provided during the annotating step (***page 5, figure 4, run-time formula to calculate C_i.max_time***).

10.6.1. Regarding (***page 5, figure 4, run-time calculation of C_i.max_time***); it would have been obvious to use the dynamic run-time formula as the annotation.

10.7. Regarding claims 20 and 52, the art of Suzuki and the art of Lajolo are analogous art because they both are directed to performance estimation for hardware/software co-design.

10.8. Regarding claims 20 and 52, the motivation to use the art of Suzuki with the art of Lajolo would have been obvious given the benefit of adaptable performance calculations provided by run-time formula evaluation.

10.9. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to produce the claimed invention.

11. Claims 27 and 59 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lajolo (Lajolo, Marcello; Lazarescu, Mihai; Sangiovanni-Vincentelli, Alberto; "A Compilation-based Software Estimation Scheme for Hardware/Software Co-Simulation", 7th International Workshop on Hardware/Software Co-Design, May 3 – 5, 1999) and Mak (Mak,Ronald; "Writing Compilers and Interpreters: an applied approach", 1991, John Wiley & Sons) in view of Suzuki (Suzuki, Kei; Sangiovanni-Vincentelli, Alberto; "Efficient Software Performance Estimation Methods for Hardware/Software Codesign", 1996, Proceedings of the 33rd annual conference on Design Automation).

11.1. Claim 27 is a dependent claim of claim 23, and thereby inherits all of the rejected limitations of claim 23.

11.2. Claim 59 is a dependent claim of claim 55, and thereby inherits all of the rejected limitations of claim 55.

11.3. The art of Lajolo is directed toward compilation-based software estimation scheme for hardware/software co-simulation (Title).

11.4. The art of Suzuki is directed to performance estimation methods for hardware/software codesign (***Title***).

11.5. Regarding claims 27 and 59, Lajolo does not specifically teach that performance information is formula for dynamically computing a value.

11.6. Regarding claims 27 and 59, Suzuki appears to teach that performance information is formula for dynamically computing a value (***page 5, figure 4, run-time formula to calculate C_t.max_time***).

11.7. Regarding claims 27 and 59, the art of Suzuki and the art of Lajolo are analogous art because they both are directed to performance estimation for hardware/software co-design.

11.8. Regarding claims 27 and 59, the motivation to use the art of Suzuki with the art of Lajolo would have been obvious given the benefit of adaptable performance calculations provided by run-time formula evaluation.

11.9. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to produce the claimed inventions.

Conclusion

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Russell L. Guill whose telephone number is 571-272-7955. The examiner can normally be reached on Monday – Friday 9:00 AM – 5:30 PM.

13. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kevin Teska can be reached on 571-272-3716. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306. Any inquiry of a general nature or relating to the status of this application should be directed to the TC2100 Group Receptionist: 571-272-2100.

14. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

RG

[illegible]

KEVIN J. TESKA
SUPERVISORY
PATENT EXAMINER